Learning Cooperative Policies with Graph Networks in Distributed Swarm Systems

Tianle Zhang, Zhen Liu, Zhiqiang Pu, Jianqiang Yi, *Senior Member, IEEE*, Xiaolin Ai, and Wanmai Yuan

***Abstract*—Deriving efficient cooperative policies in uncertain dynamic environments poses huge challenges for a distributed swarm system due to the limited capability of the agents and the complex dynamics of the environment. In this paper, a novel distributed method based on deep reinforcement learning using observation-level and communication-level graph networks is proposed to learn cooperative policies for the distributed swarm system. Specifically, a relational directed graph attention neural network is designed to model observation-level graphs composed of heterogeneous relational graphs among each agent and each type of entities (e.g., obstacles, other teammates, opponents), for extracting different relational representations. Moreover, a relevant directed graph attention network is presented to cut off the ineffective communication among irrelevant agents, and model a relevant communication topology between each agent and relevant homogeneous neighbor agents as an communication-level graph, for promoting efficient inter-agent interactions. Furthermore, a distributed actor-critic algorithm with full parameter sharing is implemented to learn cooperative swarm policies by using distributed critics, which avoids the curse of dimensionality under a centralized critic. Various simulation results validate the effectiveness and generalization of the proposed method, and demonstrate that the proposed method outperforms existing state-of-the-art methods on coverage and pursuit tasks.**

***Index Terms*—Distributed swarm systems, deep reinforcement learning, graph neural networks, multi-agent systems.**

I. Introduction

In recent years, swarm systems composed of unmanned ground or aerial vehicles have attracted more and more attention among researchers due to the unique benefits and the promising broad applications. For the benefits, they can accomplish complex tasks in a low-cost and highly decentralized form, and realize independent coordination, dynamic adjustment and self-healing combination in complex environments [1]. Their applications can be found in warehousing logistics [2], search and rescue scenarios [3], satellite clusters [4], etc. A swarm system has many homogeneous agents who need to achieve a common goal through interaction with each other in the uncertain dynamic environment. Generally, each agent in the swarm system has limited capabilities in terms of communication, sensing and manipulation, such that the required tasks need to be completed collectively by multiple agents with distributed control methods. Moreover, the agents are replaceable and interchangeable, and the number of the agents is uncertain. Therefore, due to the limited capability of the agents and the complex dynamics of the environment, it is still challenging to find cooperative policies that can enable the distributed swarm system to derive cooperative behaviors for completing specified tasks.

Recently, deep reinforcement learning (DRL) has shown great potential to solve multi-agent cooperative problems. Some DRL methods are proposed by designing global cooperation mechanisms, such as centralized critic [5], communication among agents [6], and joint value factorization [7]. However, most of these methods cannot be directly applied to swarm systems due to some challenges:

a) The high dimensionality of state and observation information caused by the increase in the number of agents;
b) Changing of the available information set size, because the number of observed neighbor entities increases or decreases over time for each agent.

Although there are the challenges, some research works with DRL for swarm systems have made good progress. A guided DRL method is proposed to learn to control a group of cooperative agents with limited sensing capabilities [8]. But, this method uses a histogram over distances with a fixed dimension size as the observation representation, which has poor scalability and generalization. Considering this drawback, a state representation method based on mean embeddings of distributions is designed to adapt to the change of information set size for swarm systems [9]. But, this method just uses a mean operation for the information of surrounding entities for each agent, and ignores natural spatial structures among each agent and its neighboring entities.

In reality, a swarm system can be naturally described as a graph structure, which can represent spatial relations among agents. Meanwhile, there may be multiple different types of entities in the environment, e.g., obstacles, other teammates, and opponents. Different types of entities have different influence relations on each agent, e.g., avoiding obstacles, cooperating with other teammates, and capturing opponents. Therefore, each agent and the entities belonging to the same

This work was supported by the National Key Research and Development Program of China under Grant 2018AAA0102402, in part by the National Natural Science Foundation of China under Grant 62073323, in part by the Strategic Priority Research Program of Chinese Academy of Sciences under Grant XDA27030204, in part by the Science and Technology Development Fund of Macau under Grant 0025/2019/AKP, and in part by the Beijing Nova Program under Grant 20220484077. *(Corresponding author: Zhen Liu.)*

Tianle Zhang, Zhen Liu, Zhiqiang Pu, and Jianqiang Yi are with School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing 100049, China, and with Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China (e-mail: tianle-zhang@outlook.com, liuzhen@ia.ac.cn, zhiqiang.pu@ia.ac.cn, jianqiang.yi@ia.ac.cn). Xiaolin Ai is with Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China (e-mail: xiaolin.ai@ia.ac.cn). Wanmai Yuan is with the Information Science Academy of China Electronics Technology Group Corporation, Beijing 100081, China (e-mail: yuanwanmai7@163.com).

type should be modeled as a graph for representing their special spatial influence relations. Besides, the communication topology among each agent and its teammates is also usually represented as a graph structure. Through this graph, agents can transmit effective messages to promote their effective interactions.

In recent years, graph neural networks (GNNs) [10], [11] are widely used and deeply researched to process graph-structured data for finding the relations among nodes in graphs. Therefore, it is a natural idea that the swarm systems can be modeled as graph structures through GNNs, and processed for extracting spatial relational representations. Similarly, the communication topology among agents can be modeled by GNNs. Currently, there are some multi-agent learning works using GNNs to deal with the observation and communication information [12]–[14]. Especially, some of them integrate attention mechanisms to enable agents to selectively focus on important information. However, most of them ignore different relations among each agent and other entities in the environment, and they do not consider the invalid inter-agent communication that would weaken the focus on truly important communication. Moreover, they do not simultaneously consider observation-level and communication-level graphs for each agent.

Motivated by the aforementioned discussions, a new distributed method based on DRL using observation-level and communication-level graphs, called **R**elational and **R**elevant directed graph **A**ttention based **S**warm actor-critic (RRAS), is proposed to learn cooperative policies for distributed swarm systems in this paper. Specifically, a relational directed graph attention neural network (RDGANet) is designed to model observation-level graphs composed of heterogeneous relational graphs among each agent and each type of entities (e.g., obstacles, other teammates, opponents), for extracting different relational representations. Moreover, a relevant directed graph attention network (RGAT) is presented to cut off the ineffective communication between irrelevant agents, and model a relevant communication topology among each agent and relevant homogeneous neighbor agents as an communication-level graph, for promoting efficient inter-agent interactions. Furthermore, to avoid the curse of dimensionality under a centralized critic, a distributed actor-critic algorithm with full parameter sharing is implemented to learn cooperative swarm policies.

II. Background

A. Distributed Swarm System

In this paper, a distributed swarm system shown in Fig. 1, where n homogenous and autonomous agents (green circles) try to carry out tasks together, is studied in a two-dimensional space [15]. Meanwhile, there are other entities (black circles) in this environment, e.g., obstacles and opponents. For simplicity, we assume that the geometry of the agents and other entities is modeled as a disc with a radius. For clarity, we refer to the agents and other entities as entities. The state $s_i = [p_i, v_i]$ of agent i ($i \in \{1, ..., n\}$) includes its own position $p_i = [p_i^x, p_i^y]$ and velocity $v_i = [v_i^x, v_i^y]$, which can be observed by other agents. Each agent is partially observable, and can only obtain its own state and the states of neighboring entities (e.g., obstacles, other teammates and opponents) within its observation scope. The state of the neighbor entity consists of its position and velocity. The observation scope of each agent is a circular area with radius D^o. Meanwhile, each agent has only local communication capability, and can only communicate with other homogenous agents (i.e., teammates) within its communication scope. The communication scope is a circular area with radius D^c. Beside, the dynamic model of each agent is modeled as a double integrator. The action of agent i can be expressed as $a_i = [F_i^x, F_i^y]$, where F_i^x, F_i^y represent the forces applied to the agent in both directions. In the swarm system, the number of entities within the observation scope of agent i and the number of homogeneous agents in the communication scope of the agents will change dynamically at each timestep. Therefore, agent i needs to make efficient decisions based on its partial observations and local communication messages in uncertain dynamic environments. In this paper, we study how to learn cooperative policies for the distributed swarm system to accomplish specified tasks efficiently.

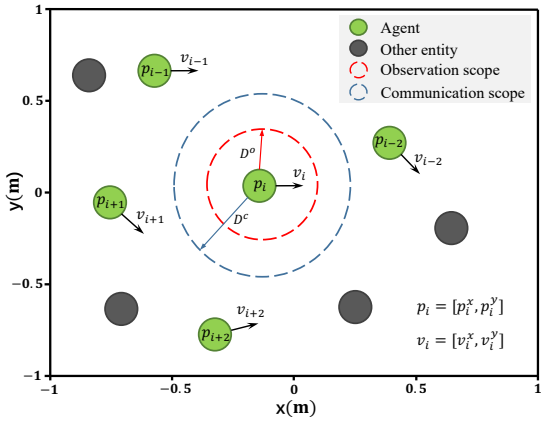


Fig. 1: Illustration of the distributed swarm system.

B. Partially Observable Markov Games

A multi-agent extension of Markov decision processes, called partially observable Markov game [16], is considered for the distributed swarm system. A Markov game for n agents is defined through: a set of states $\mathcal{S}$ describing all possible global state of the game, a set of actions $\mathcal{A}_1, ..., \mathcal{A}_n$ and a set of partial observations $\mathcal{O}_1, ..., \mathcal{O}_n$ that the agents can acquire. In the process of environment interaction, each agent i uses a policy $\pi_i : \mathcal{O}_i \mapsto \mathcal{A}_i$, to choose an action $a_i \in \mathcal{A}_i$ based on its partial observation $o_i \in \mathcal{O}_i$. Meanwhile, the environment state evolves to the next state according to the state transition function $\mathcal{T} : \mathcal{S} \times \mathcal{A}_1 \times ... \times \mathcal{A}_n \mapsto \mathcal{S}$. Each agent i can obtain a reward R_i by a reward function: $\mathcal{S} \times \mathcal{A}_1 \times ... \times \mathcal{A}_n \mapsto \mathbb{R}$, and perceives a private partial observation $o_i \in \mathcal{O}_i : \mathcal{S} \mapsto \mathcal{O}_i$. The initial state is determined through a state distribution $\rho : \mathcal{S} \mapsto [0, 1]$. The goal of each agent i is to maximize its

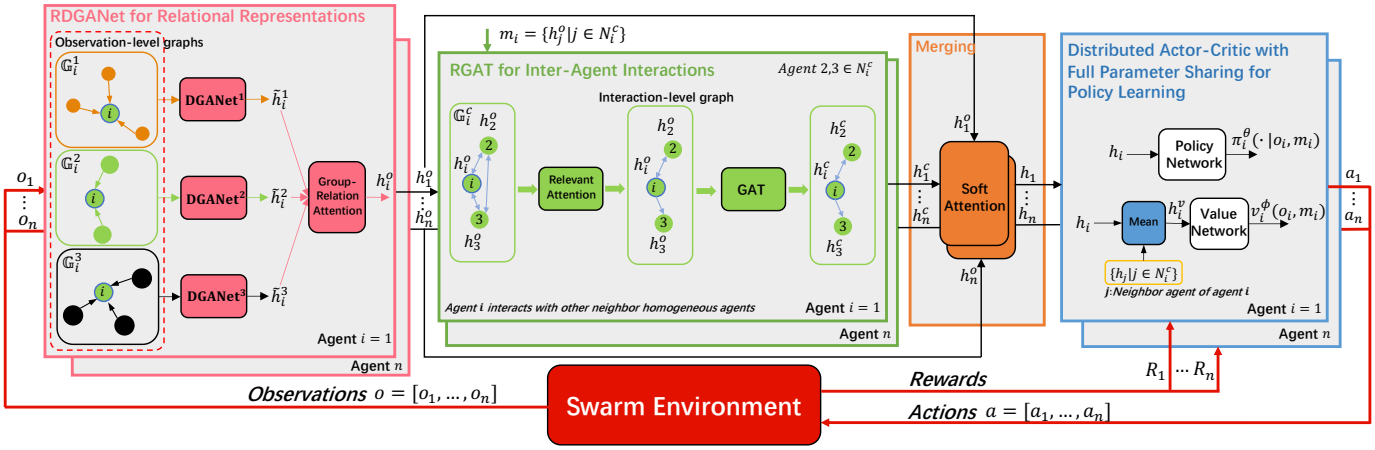


Fig. 2: Overall Structure of RRAS.

own expected cumulative return $\mathbb{E}[\sum_{t=0}^{T} \gamma^t R_i(t)]$ over a certain period of time, where $\gamma \in [0, 1]$ is a discount factor and T is the time horizon.

III. Method

A. Overall Structure of RRAS

A new distributed method using observation-level and communication-level graphs based DRL, named RRAS, is formally proposed to learn cooperative policies for the distributed swarm system. As shown in Fig. 2, the overall structure of the proposed RRAS consists of four main modules. 1) RDGANet is designed to model observation-level graphs composed of heterogeneous relational graphs among each agent and each type of entities (e.g., obstacles, other teammates, opponents) through using multiple DGANets, for extracting different relational representations. Then, a group-relation attention network is adopted to selectively centralize different influence relations from different relational graphs. 2) RGAT is specially presented to firstly cut off the ineffective communication between unrelated agents by using a relevant attention mechanism, and model a relevant communication topology among each agent and relevant homogeneous neighbor agents as an communication-level graph by using GAT, for facilitating efficient inter-agent interactions. 3) Merging module with a soft attention mechanism is given to effectively merge observation and interaction embeddings by complementing each other, for boosting the environmental representation of each agent. 4) Distributed actor-critic algorithm with full parameter sharing is implemented to learn cooperative swarm policies by using distributional critics, which avoids the state dimension disaster under a centralized critic. In the following, each module will be described in detail.

B. RDGANet for Different Relational Representations

In a complex swarm environment, there are multiple different types of entities in the observation scope of each agent, e.g., obstacles, other teammates, opponents. Each agent has different relations with different types of entities, such as cooperation with teammates, competition with opponents, avoidance with obstacles and so on. Through representing these different relations, each agent can understand its surrounding environment at a higher level, and then selectively focus on one of the relations that is beneficial to accomplishing tasks. Besides, graph attention neural networks can be used to effectively represent natural spatial relations among entities, and the attention mechanisms can enable each agent to selectively focus on important entities through assigning attention weights for efficiently completing the specific task [17], [18]. Meanwhile, Transformer [19] is a new simple network architecture based solely on attention mechanisms with scaled dot-product attention. Since the scaled dot-product attention in Transformer is invariant to the number and permutation of nodes, it shows great potential in the field of multi-agent systems for effectively assigning attention weights among agents [12], [20]. Inspired by the above, we design RDGANet based on Transformer to model observation-level graphs composed of heterogeneous relational graphs among each agent and each type of entities, for extracting different relational representations. The modeling process of RDGANet consists of the following three stages.

1) Group Division: The first stage is to cluster all entities into K groups according to different types of entities using prior knowledge. The same type entities are clustered into one group, where the k-th group is denoted with C^k, $k = 1, ..., K$. For instance, in a predator-prey game, all predators can be categorized into one group, and all preys are clustered into another group. Besides, the entities that do not perform any actions but participate in the game can be clustered into a group, e.g., obstacles or targets. According to the divided groups, the partial observation of agent i can be represented as $o_i = \{o_i^1, ..., o_i^K\}$, where $o_i^k = \{s_j^k | j \in N^o(i)\}$, $s_j^k = [v_j^x, v_j^y, p_j^x, p_j^y]^T$ is the observation state of entity j, k is the index of the group that entity j belongs to, and $N^o(i)$ is a set of neighborhoods (include itself) within the visual range of agent i.

2) Relational Graph Modeling: Then, multiple directed graph attention neural networks (DGANets) based on Transformer, which have the same structure but different parameters, are adopted to model heterogeneous relational graphs for dif-

ferent groups in the partial observation of agent i. A relational directed graph $\mathbb{G}_i^k := (\mathbb{V}_i^k, \mathbb{E}_i^k)$ is modeled by a DGANet based on agent i's partial observation o_i^k. In this graph, each node is either agent i or other neighboring entity j belonging to k group, $j \in C^k \cap N^o(i)$, and there exist directed edges pointing to agent i from the neighboring entities. In the following, the modeling process of a DGANet (e.g., $DGANet^k$) is described in detail through a simple scene where there are three entities. As shown in Fig. 3, agent i can observe its neighboring entities belonging to the k-th group, i.e., entity 2 and entity 3. Based on this scene, a relational directed graph can be modeled by a DGANet for agent i, where nodes are the three entities and there exist edges from entities 2 and 3 to agent i. Specifically, the observation state of entity j, $j \in C^k \cap N^o(i)$, is firstly encoded as the feature embedding,

$$h_j^k = \begin{cases} \boldsymbol{W}_S \cdot s_i^k & j = i \\ \boldsymbol{W}_N^k \cdot (s_j^k - s_i^k) & j \neq i \end{cases}, \tag{1}$$

where $\boldsymbol{W}_S, \boldsymbol{W}_N^k$ are learnable parameter matrices. Now, entity j computes a key $E_j^k = \boldsymbol{W}_E^k h_j^k$, query $Q_j^k = \boldsymbol{W}_Q^k h_j^k$ and value $V_j^k = \boldsymbol{W}_V^k h_j^k$ vectors where $\boldsymbol{W}_E^k, \boldsymbol{W}_Q^k, \boldsymbol{W}_V^k$ are other learnable parameter matrices. Then, after receiving query-value pair (Q_i^k, E_j^k) from its neighbors, agent i assigns weights to corresponding neighbors:

$$\alpha_{ij}^k = softmax(\frac{Q_i^{kT} E_j^k}{d_E}), \tag{2}$$

where d_E is the dimensionality of the key vector. Next, it computes the aggregated embedding according to the assigned weights:

$$\hat{h}_i^k = \boldsymbol{W}_{out}^k \sum_{j \in G^k \cap N(i)} \alpha_{ij}^k V_j^k, \tag{3}$$

where $\boldsymbol{W}_{out}^k$ is another learnable parameter. Finally, agent i calculates the final relational embedding $\tilde{h}_i^k$ by doing a non-linear transformation of an embedding composed of h_i^k concatenated with $\hat{h}_i^k$ by using a one-FC-layer network $\mathbb{F}$, i.e., $\tilde{h}_i^k = \mathbb{F}([h_i^k \| \hat{h}_i^k])$. Herein, the relational embedding $\tilde{h}_i^k$ of the k-th group to agent i can be obtained, which implicitly represents the spatial influence relation.

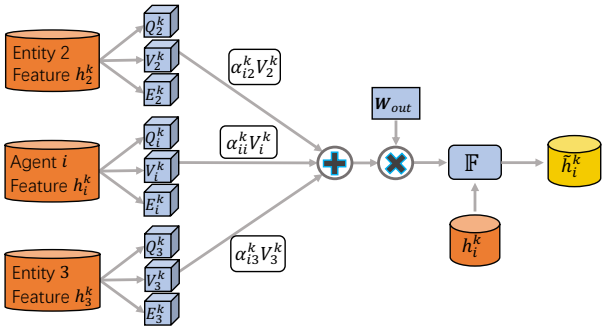


Fig. 3: Illustration of DGANet.

3) Attention Aggregation: Finally, to enable agent i to selectively centralize different influence relations from different groups, we design a group-relation attention mechanism to aggregate the relational embeddings as shown in Fig. 4. In this mechanism, an embedding concatenating the relational embedding $\tilde{h}_i^k$ and the encoding embedding h_i^k is fed into a two-FC-layer network $\mathcal{F}_1$, which outputs a coefficient value a_i^k. Based on different coefficient values, each relational embedding is assigned a weight $\beta_i^k = softmax(a_i^k)$, $k = 1, ..., K$. Finally, the observation embedding of agent i can be obtained by weighted summation as follow:

$$h_i^o = \sum_{k=1}^{K} \tilde{h}_i^k * \beta_i^k. \tag{4}$$

Here, the observation embedding h_i^o of agent i is got by RDGANet, which implicitly encodes the spatial influence relation of the surrounding entities on agent i.

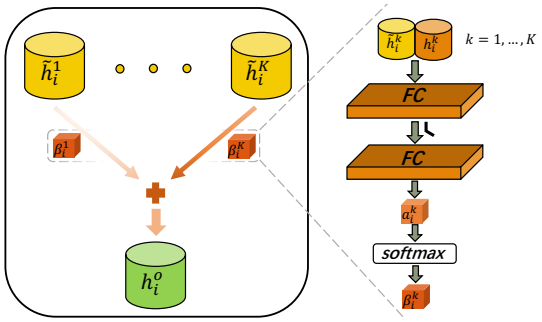


Fig. 4: Group-relation attention mechanism.

C. RGAT for Efficient Inter-Agent Interactions

Effective communication among homogeneous agents can not only enlarge the receptive field of each agent, but also promote cooperation. In reality, the communication topology of a swarm system is always represented as a graph, and each edge of the graph conveys a different communicate message. However, if all neighboring agents in the swarm system communicate with each other, it will lead to data disaster and communication delay, yielding inefficient interactions. Meanwhile, there is a lot of useless communication inside, which weakens attention given to the few truly significant communication among agents [21]. Therefore, we design RGAT to firstly cut off the ineffective communication between irrelevant agents, and model a relevant communication topology among each agent and relevant homogeneous neighbor agents as an communication-level graph. RGAT consists of the following two modules:

- Relevant Attention: Remove the edges between irrelevant neighbor agents in a full connected communication topology, yielding a relevant communication topology.
- Graph Attention Network (GAT) [11]: Model the constructed relevant communication topology as an communication-level graph and extract effective interaction messages from the graph for each agent.

The modeling process of RGAT consists of the following three steps.

1) Firstly, we define a fully connected communication topology $\mathbb{G}_i^c := (\mathbb{V}_i^c, \mathbb{E}_i^c)$ for agent i, where each node $r \in \mathbb{V}_i^c$

denotes agent i or other homogeneous agents within the communication scope of the agent, and there exists an edge $e \in \mathbb{E}_i^c$ between two nodes if the distance between the nodes is less than the communication distance D^c. The observation embedding h_i^o of agent i is used as the interactive message to be transmitted in this graph. Therefore, agent i can receive a local interaction message set $m_i = \{h_j^o | j \in N_i^c\}$ from other neighbor homogeneous agents, where N_i^c is a set of other homogeneous agents (include itself) within the communication scope of agent i.

2) After that, the edges between irrelevant neighbor agents in the graph are removed through a relevant attention mechanism as shown in Fig. 5. Agents 2 and 3 are the neighbors of agent i, i.e., $2, 3 \in N_i^c$. For simplicity, $j \in N_i^c$ denotes the neighbor of agent i. The relevant attention mechanism is adopted to learn the relevant weight $w_{i,j}^r$, which is a scalar value (0 or 1) and determines whether there exists interaction relation between agents i and j in the swarm system. Specifically, firstly, the interaction relations among the agents need to be learned. The output of the traditional Long Short-Term Memory (LSTM) network only depends on the inputs of the current time and the previous time, which does not take advantage of the information of all times. On the contrary, Bi-directional Long Short-Term Memory (BiLSTM) network can utilize the inputs at all times, which is a great attribute that deliberates the relations of all inputs. Therefore, the BiLSTM network is adopted to learn interaction relations among the agents in this paper. In particular, we concatenate the observation embeddings of agent i and agent j as an embedding, where $j \in N_i^c$ and $j \neq i$. This embedding is fed into the BiLSTM network:

$$\{h_{i,j}\} = f(BiLSTM(\{[h_i^o \| h_j^o]\})), \tag{5}$$

where $f(\cdot)$ is a FC layer network, $h_{i,j}$ is a two-tuple vector that indicates related and unrelated classes. Then, since sampling based on softmax function is not able to realize back-propagation of gradients, the gumbel-softmax function is adopted for obtaining the relevant weight,

$$w_{i,j}^r = gum(h_{i,j}), \tag{6}$$

where $gum(\cdot)$ is the gumbel-softmax function and $w_{i,j}^r$ is 0 or 1. $w_{i,j}^r = 0$ and $w_{i,j}^r = 1$ denote relevant and irrelevant interaction relations between agents i and j respectively. Meanwhile, since agent i is related to itself, $w_{i,i}^r$ is set to 1. The relevant weight $w_{i,j}^r$ indicates whether there is a directed edge from agent j to agent i in the graph $\mathbb{G}_i^c$. By the relevant attention mechanism, the directed edges between irrelevant agents in the communication topology are removed, yielding a relevant communication topology.

3) Finally, GAT is implemented to model the relevant communication topology as an communication-level graph and extract interaction messages from this graph. In particular, the observation embeddings are treated as interaction messages among the agents in the graph. The observation embeddings only represent each agent's understanding of the surrounding environment, without containing cooperative information.

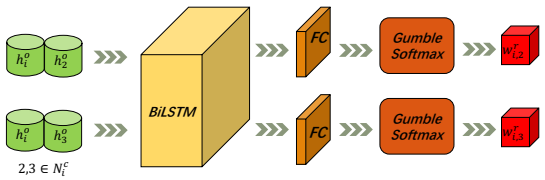


Fig. 5: Relevant attention mechanism.

Now, after agent i receives the local interaction messages $m_i = \{h_j^o | j \in N_i^c\}$, these interaction messages are transformed to encodings, i.e., $E_j = \boldsymbol{W}_c h_j^o$, where $\boldsymbol{W}_c$ is a learnable parameter matrix. Based on E_j, the attention coefficient can be computed as follow:

$$\xi_{ij} = LeakyReLU(\boldsymbol{a}^T [w_{i,j}^r E_i \| w_{i,j}^r E_j]), \tag{7}$$

where $\boldsymbol{a}^T$ is a learnable parameter vector, $LeakyReLU$ is a nonlinear activation function and $j \in N_i^c$. Based on the attention coefficient, the attention weight between agents i and j is calculated,

$$c_{ij} = \begin{cases} \frac{exp(\xi_{ij})}{\sum_{r \in N_i^c} exp(\xi_{ir})} & if \quad A_{ij} = 1 \\ 0 & otherwise \end{cases}, \tag{8}$$

where $\boldsymbol{A} = \{A_{ij}\}$ is the adjacency matrix of the graph. If two agents are within their respective communication scope, $A_{ij} = 1$. Otherwise, $A_{ij} = 0$. Then, agent i aggregates all effective interaction messages through computing a weighted sum of its neighbors' encoding, yielding the interaction embedding $h_i^c = \sum_{j \in N_i^c} c_{i,j} E_j$. Hence, the interaction embeddings are obtained by RGAT, which implicitly represent effective interaction messages among each agent and its collaborators.

D. Merging Observation and Interaction Embeddings

Considering the respective problems of partial observation and local communication in the real world: the incompleteness and inaccuracy of the observation information, unreliable interaction caused by communication delay, we combine the observation and interaction embeddings to complement each other. Meanwhile, in order to enable agents to effectively merge both cooperation embeddings and observation embeddings, we use a soft attention mechanism. In this mechanism, a embedding concatenating h_i^o and h_i^k is fed into a two-FC-layer network $\mathcal{F}_2$, which outputs an observation coefficient value a_i^o. Similarly, a embedding concatenating h_i^c and h_i^k is fed into the network $\mathcal{F}_2$, which outputs an interaction coefficient value a_i^c. After getting the coefficient values, the observation and interaction embeddings are assigned the weights $\zeta_i^o = softmax(a_i^o) = \frac{exp(a_i^o)}{exp(a_i^o)+exp(a_i^c)}$ and $\zeta_i^c = softmax(a_i^c) = \frac{exp(a_i^c)}{exp(a_i^o)+exp(a_i^c)}$, respectively. Subsequently, the environmental embedding h_i is defined by weighted sum using the weights,

$$h_i = h_i^o * \zeta_i^o + h_i^c * \zeta_i^c. \tag{9}$$

Here, the environmental embedding h_i is a state representation of the surrounding environment of agent i, which implicitly represents agent i's understanding of the surrounding environment.

E. Distributed Actor-Critic with Full Parameter Sharing for Policy Learning

The framework of centralized training with decentralized execution (CTDE) [5] is commonly adopted to learn centralized critics to update the decentralized policies during training. During testing, only the decentralized policies are executed. However, the centralized critic easily causes the state dimension disaster as the number of agents increases and cannot be transferred to new environments. Therefore, in this paper, we design a distributed actor-critic framework with full parameter sharing for policy learning, where distributed critics are learned to update distributed actors by allowing reward functions to use extra information. Specifically, based on the descriptions in the above subsections B, C, D, the environmental representation network composed of RDGANet, GAT and soft attention mechanism is firstly defined, which takes the partial observation o_i of agent i as input and outputs the environmental embedding h_i. Then, the actor and critic of agent i are defined as follow ($i = 1, ..., n$),

1) Agent i's actor parameterized by θ, composed of environmental representation network and policy network, takes the partial observation o_i and local interaction message set m_i of agent i as input and outputs action values of agent i for making decisions, i.e., $\pi_i^\theta : o_i \times m_i \mapsto a_i$.
2) Agent i's critic parameterized by ϕ, composed of environmental representation network and value network, takes the partial observation o_i and local interaction message set m_i of agent i as input and outputs a scalar value for guiding the actor training, i.e. $v_i^\phi : o_i \times m_i \mapsto \mathbb{R}$.

In the actor, the environmental embedding h_i is fed into a policy network $\mathcal{P}(\cdot)$ composed of two FC layers, which outputs the action values. In the critic, the mean of the environmental embeddings of agent i and its communication neighbors, i.e., $h_i^v = mean(\sum_{j \in N_i^c} h_j)$, is fed into a value network $\mathcal{V}(\cdot)$ composed of two FC layers, which outputs a scalar value.

Moreover, the distributed actor-critic framework adopts full parameter sharing for efficient training. Firstly, each actor and critic share the environmental representation network, including RDGANet, GAT and soft attention mechanism. Secondly, n homogeneous agents share all the learnable parameters of the actors and critics, including those of environmental representation network, policy network, and value network. Since each agent receives different observations, obtains interaction messages from other agents differently and perceives the environment differently (relative state of all the entities), sharing parameters does not prevent them from producing different behaviors.

Furthermore, as shown in Algorithm 1, an novel algorithm based on proxima policy optimization (PPO) [22] is used to efficiently train the actors and critics for generating effective cooperative policies. This algorithm is different from the traditional PPO algorithm. Firstly, multiple threads are opened to simulate the interactions between the agents and the swarm environment in parallel. Fusion experiences from multiple parallel environments are collected to train the actors and critics, which can speed up the process of the training. Then, since the actors and critics share the environmental representation network, a total loss L is used to update all the learnable network parameters, instead of updating the actors or the critics individually. The total loss is defined by the weighted summation of a value loss $L_{v_i^\phi}$, an action loss $L_{\pi_i^\theta}$ and an action entropy term H. Herein, the action entropy term is specially designed to encourage exploration for the agents by penalizing the entropy reduction of the actors.

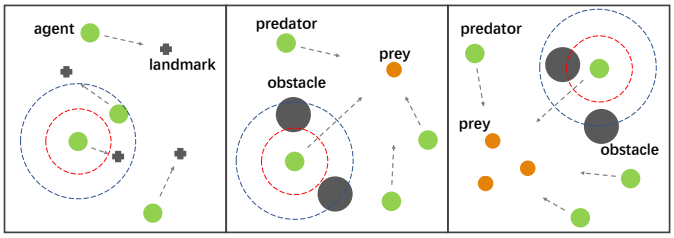
Algorithm 1 Training Algorithm of RRAS

Initialize the network parameters of actor π_i^θ and critic v_i^ϕ, i.e., RDGANet, RGAT, soft attention mechanism, policy and value networks parameters
Build W parallel environments
for Update number =1 to M **do**
  Collect training samples $\{o_i, m_i, a_i, R_i\}$ from W parallel environments by running an actor π_i^θ with T time-steps in each environment. The actor selects different actions a_i based on different observation information o_i and interaction messages m_i ($i = 1, ..., n$).
  Compute state-action values $Q_i(t) = \sum_{t'>t}^{T} \gamma^{t'-t} R_i(t)$
  Compute advantage estimate values $\hat{A}_i(t) = Q_i(t) - v_i^\phi(t)$. a critic v_i^ϕ outputs judgement values v_i for the selected actions a_i based on observation information o_i and interaction messages m_i
  Store memory $\mathcal{D} = \{o_i, a_i, R_i, Q_i, \hat{A}_i\}$, $i = 1, ..., n$
  $\pi_i^{\theta_{old}} \leftarrow \pi_i^\theta$
  for $Epoch$ = 1 to 4 **do**
    Sample training samples $\mathcal{L}$ from memory $\mathcal{D}$
    Calculate value loss: $L_{v_i^\phi} = \frac{1}{|\mathcal{L}|} \sum_{\tau \in \mathcal{L}} (Q_i - v_i^\phi)^2$
    Calculate action loss: $L_{\pi_i^\theta} = -\frac{1}{|\mathcal{L}|} \sum_{\tau \in \mathcal{L}} min(\frac{\pi_i^\theta}{\pi_i^{\theta_{old}}} \hat{A}_i, clip(\frac{\pi_i^\theta}{\pi_i^{\theta_{old}}}, 1-\epsilon, 1+\epsilon) \hat{A}_i)$
    Calculate total loss: $L = \psi_1 L_{v_i^\phi} + \psi_2 L_{\pi_i^\theta} - \psi_3 H$, $H = -\sum \pi_i^\theta log(\pi_i^\theta)$
    Update actor and critic network parameters by minimizing L with gradient descent algorithm
  end for
  Clear memory $\mathcal{D}$
end for

IV. Simulations

In order to evaluate the performances of the proposed RRAS, we design three kinds of simulation tasks as shown in Fig. 6: coverage, pursuit with a single prey and pursuit with multiple preys. The coverage and pursuit tasks are fully cooperative and mixed cooperative-competitive tasks, respectively. The tasks are built on the multi-agent particle environment (MAPE) [5]. The environment is set to be 2×2 square units in two-dimensional space where the agents adopt a double integrator dynamics model. The action space of each agent is discrete, and the agent can accelerate and decelerate in X and Y directions. In addition, the initial positions of all

entities (agents, obstacles, landmark, prey, etc.) are randomly placed. At each timestep, each agent observes the local state in its field of view: its own velocity and position, the relative distance between it and other entities, and interacts with other neighbor agents, and then, selects one of the following actions: accelerate to north, south, east or west, and no acceleration. Next, each agent can receive a team reward that is defined by the relative distance between it and other entities.



(a) Coverage (b) Pursuit with a single prey (c) Pursuit with multiple preys

Fig. 6: Illustrations of three simulation tasks. Red and blue dotted circles represent the observation scope and communication scope of each agent, respectively.

In the training phase, the network parameter settings of the proposed method are shown in TABLE I. Meanwhile, the training hyperparameters are set: $\gamma = 0.99$; $\psi_1 = 1$; $\psi_2 = 0.5$; $\psi_3 = 0.01$; $\epsilon = 0.2$. After accumulating experiences for total 4096 timesteps which are 128 timesteps on $W = 32$ parallel environments, each update of all network parameters is implemented with 4 epochs through the algorithm as shown in Algorithm 1. The evaluation is carried out after every 50 updates on 100 episodes with a new seed environment. In this phase, each agent chooses greedy discrete actions. Besides, in the training or evaluation phase, when one of the following two conditions is satisfied, the episode will be terminated. The first condition is that the task is successfully accomplished. Another one is that the episode reaches 50 timesteps. Besides, curriculum learning [23] is adopted to speed up the process of learning cooperative policies.

TABLE I: Network parameter settings of the proposed method.

Parameter	Value or Size
D^o, D^c	1.5, 2.0
$BiLSTM$ input and hidden size	256, 128
$\boldsymbol{W}_S$, $\boldsymbol{W}_N^k$	$\mathcal{R}^{128X4}$, $\mathcal{R}^{128X4}$
$\boldsymbol{W}_E^k$, $\boldsymbol{W}_Q^k$, $\boldsymbol{W}_V^k$	$\mathcal{R}^{128X4}$, $\mathcal{R}^{128X4}$, $\mathcal{R}^{128X4}$
$\boldsymbol{W}_{out}^k$, $\boldsymbol{W}_c$, $\boldsymbol{a}^T$	$\mathcal{R}^{128X128}$, $\mathcal{R}^{128X128}$, $\mathcal{R}^{256X1}$
$f(\cdot)$, $\mathbb{F}(\cdot)$	(128, 2), (256, 128)
$\mathcal{F}_1(\cdot)$	(256, 128, 128, 1)
$\mathcal{F}_2(\cdot)$	(256, 128, 128, 1)
$\mathcal{P}(\cdot)$	(128, 128, 128, 5)
$\mathcal{V}(\cdot)$	(128, 128, 128, 1)

Three state-of-the-art methods are selected as baseline methods: 1) **LTCB** [12]: the method uses graph attention network to process the local observation and communication information of the agent, but does not group the local observation and remove redundant communication; 2) **DyAN** [13]: the method uses graph neural network to deal with the local dynamic observation of the agent, and does not involve the attention mechanism; 3) **AERL** [14]: the method uses graph attention network and LSTM to promote communication among agents. Furthermore, we use 3 metrics to evaluate different methods: **Success Rate**(S%): Percentage of episodes the agents completed the task. **Mean episode reward** (MER): Mean of episode reward for each agent. **Mean episode length** (MEL): Mean of successful episode length. In addition, the update number (UN) of the network parameters also needs to be noted, i.e., M shown in Algorithm 1.

A. Coverage

1) Task Description: In a coverage task, n homogeneous agents are required to cooperatively reach n landmarks without colliding with each other. The agents have to cover all of the landmarks. There is no specific matching relationship between the landmarks and the agents, which means that the landmarks are not assigned to each agent. Each agent need to learn to observe the effective information of the landmarks and other agents within it visual area, and interact with collaborators for covering all landmarks. In the coverage task, there are two groups in the environment, i.e., landmarks and agents ($K = 2$).

2) Results: To evaluate the performance of the proposed method in swarm systems, a coverage task where 20 agents cover 20 landmarks is conducted. Since it is difficult for the agents to directly learn the cooperative policies for completing this task, curriculum learning is implemented to speed up the learning. The curriculum learning and test performances of different methods in the coverage task are shown in TABLE II. For curriculum learning, a curriculum strategy with the increasing numbers of agents and landmarks is designed. Specifically, a coverage policy is firstly trained with a 3-agent team ($n = 3$). Once the evaluation success rate of this policy reaches a threshold (90%), the learned policy is directly transferred to a team with more agents to continue training. The process is repeated until the number of the team reaches 20 agents. The results show that RRAS has faster convergence with a higher success rate (97.8%) compared with other methods. On the contrary, DyAN fails due to the lack of communication and attention mechanism. RRAS outperforms LTCB and AERL. This may be because the designed RDGANet can extract different relational representations from agents' observations, and RGAT can remove invalid communication and promote effective communication. In general, the results in the coverage task fully demonstrate the effectiveness of the proposed method.

In order to further verify the generalization of the proposed method in the coverage task, the learned policy in the 20-agent team is directly tested without any fine-tuning in new scenarios with different numbers of agents. The corresponding results are shown in Fig. 7. The results demonstrate that the learned policy obtained by the proposed RRAS shows the best

TABLE II: Curriculum learning and testing performance of our method and other comparison methods in the coverage task.

	Method	n = 3		n = 6		n = 9		n = 12		n = 15		n = 18		n = 20	
Curriculum Learning		S%	UN	S%	UN	S%	UN	S%	UN	S%	UN	S%	UN	S%	UN
	LTCB [12]	94	2650	92	2850	91	4650	93	1800	91	4950	93	3450	89	6200
	DyAN [13]	93	800	94	250	92	500	93	200	98	200	0	12200	0	6200
	AERL [14]	92	2900	94	3850	93	2600	93	4300	91	3450	92	12000	90	6200
	RRAS (Ours)	95	850	93	400	93	550	97	500	91	500	92	1800	**97**	**6200**

	Method	S%	MER	MEL
Testing (n = 20)	LTCB	88.6 ± 0.21	-2.593 ± 0.0533	24.46 ± 0.214
	DyAN	0.0 ± 0.00	-4.800 ± 0.0650	50.00 ± 0.000
	AERL	90.6 ± 0.47	-2.743 ± 0.0235	26.12 ± 0.142
	RRAS (Ours)	**97.8** ± 0.12	**-1.894** ± 0.0125	**17.87** ± 0.260

generalization than other methods in the term of success rate. This indicates that the proposed method can be transferable to different team sizes while maintaining good performance. Moreover, since the policy is obtained through the curriculum learning with adding new agents until 20 agents, teams with less than 20 agents have better success rates than teams with more than 20 agents. Furthermore, in more than 20 agents scenarios that do not appear in the learning process, RRAS still achieves better generalization performance than other methods.

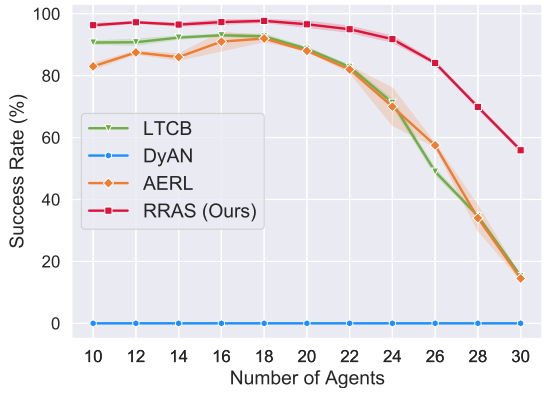
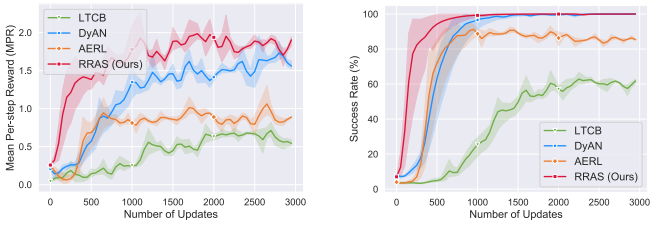


Fig. 7: Generalization performance of different methods in the coverage task.

B. Pursuit with A Single Prey

1) Task Description: In a pursuit with a single prey task, n slower predators need to cooperatively chase one faster prey around a randomly generated environment with l large static obstacles obstructing the way. Only when the cooperative predators have collided with the prey, the chasing process is completed. Meanwhile, during the chasing process, each predator needs to avoid collisions with obstacles and other predators. Therefore, each predator needs to learn to extract effective information within its visual area and cooperate with other neighbor predators for pursuing the prey while avoiding collisions. The maximum velocities of the prey and predators are 1.5 and 1.0 ($unit \cdot s^{-1}$) respectively. The maximum accelerations of the prey and predators are 4 and 3 ($unit \cdot s^{-2}$) respectively. Thus, the movement ability of the prey is obviously stronger than that of the predators. This makes the task more difficult to be completed, which requires a higher level of cooperation among the predators. Besides, two static obstacles are placed into the environment (i.e., $l = 2$). The prey adopts the fixed policy [24].

2) Results: For pursuit with one fixed-policy prey, we conduct a task scenario where ten predators chase one prey. In this task, the predators need to learn cooperative chasing policies. The training results are shown in Fig. 8, which show that RRAS have better performance than the other methods in terms of convergence rate. Furthermore, we evaluate the effectiveness of the policy obtained through the training, and the results are shown in TABLE III. The evaluate results show that the proposed RRAS has best performance compared with the other methods. the predators performing the RRAS method can catch the prey faster than the other methods, along with higher rewards. This is largely due to RDGANet, which can model the relations between each predator and the prey and implicitly predict the strategy and intention of the prey. In addition, the generalization results with different number of predators are shown in Fig. 9. The results show that RRAS has always maintained a high success rate, even when the number of predators is low. Therefore, these results fully reflect the advantages of our proposed method.



(a) Mean per-step reward (b) Success rate

Fig. 8: Training results for the predators in the pursuit task where ten predators chase one fixed-policy prey

TABLE III: Evaluation results.

Method	S%	MER	MEL
LTCB	60.5 ± 1.50	21.833 ± 2.1129	35.22 ± 0.574
DyAN	99.5 ± 0.50	23.295 ± 1.3977	15.53 ± 0.049
AERL	86.0 ± 2.00	24.028 ± 0.9266	26.55 ± 0.120
RRAS (Ours)	**100.0** ± 0.00	**28.269** ± 1.1358	**12.62** ± 0.270

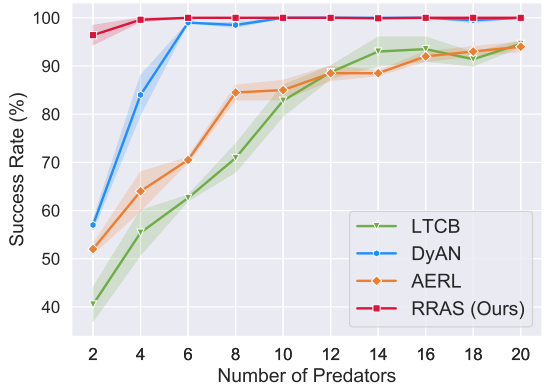


Fig. 9: Generalization performance of the different methods in the pursuit task with one fixed-policy prey.

C. Pursuit with Multiple Preys

1) Task Description: In a task of pursuing multiple preys, n slower predators need to cooperatively chase p ($p < n$) faster preys. They are all in a randomly generated environment with l large static obstacles. The goal of the predators is to capture all preys, and the preys need to stay away from the predators as much as possible. When any one of the preys is captured by the predators, it dies and keeps stationary. This chasing process continues until all preys are captured. The simulation environment of the task is similar to that of the task of pursuing a single prey. Differing from chasing one prey, the predators need to know whether each prey is dead or alive to decide whether to pursue or not. Meanwhile, each prey also need to know the life state of its partner. Hence, we give each agent j a life state s_j^{live}, which $s_j^{live} = 1$ represents the state of being alive and $s_j^{live} = 0$ represents the state of being dead for agent j. The observation state of each agent j is changed to $s_j = [v_j^x, v_j^y, p_j^x, p_j^y, s_j^{live}]$.

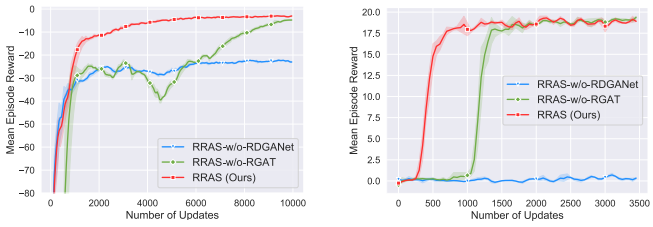
2) Results: To test the performance of our method in the pursuit task with multiple preys, we built a scenario with 20 predators, 5 preys and 2 static obstacles. In this scenario, each predator need to choose cooperation with other predators or chasing preys individually. Meanwhile, when chasing preys, it also need to choose which prey to chase. In order to form multiple comparative simulation, the predators perform the different methods to chase the preys that implement the different methods. The predators and preys need to learn chasing and escaping policies in mutual competition, respectively. The simulation results are shown in TABLE IV. This results are the mean and standard deviation of episode reward obtained for the predators under different methods. Compared with the other methods, the predators trained by RRAS achieve the highest episode reward. Similarly, the preys trained by RRAS have the best defense against the predators. We analyze that the superior performance of RRAS is due to the designed RDGANet for learning relational representations that implicitly represent the strategy and intention of the preys and the designed RGAT for promoting effective interactions among collaborators. Moreover, our method RRAS merges observation and communication information using an attention mechanism, which enables each predator to selectively attend cooperation with other predators or chase preys individually. Through the pursuit task with multiple preys, the effectiveness and advantage of our method are fully demonstrated.

TABLE IV: The mean and standard deviation of episode reward for the predators in the task of pursuing multiple preys.

Method of predators (n = 20)	Method of preys (p = 5)			
	LTCB	DyAN	AERL	RRAS
LTCB	55.04 ± 3.816	0.00 ± 0.000	55.08 ± 2.052	0.00 ± 0.000
DyAN	57.479 ± 1.948	8.00 ± 0.600	58.792 ± 1.691	6.00 ± 0.250
AERL	51.37 ± 1.043	0.00 ± 0.000	55.22 ± 3.935	0.00 ± 0.000
RRAS	**61.96** ± 3.281	**12.50** ± 1.250	**58.98** ± 0.918	**57.06** ± 1.948

D. Ablation Studies

To demonstrate the effectiveness of the main components of the proposed RRAS, i.e., RDGANet and RGAT, ablation studies regarding the main components are implemented in two difficult tasks: 8-agent coverage task and 3 vs. 1 pursuit task with one fixed-policy prey. The training results (mean episode reward vs. number of updates) are shown in Fig. 10. RRAS-w/o-RDGANet and RRAS-w/o-RGAT are ablation methods that remove RDGANet and RGAT modules on RRAS, respectively. The results demonstrate that RRAS outperforms RRAS-w/o-RDGANet and RRAS-w/o-RGAT. This fully validates the necessity and effectiveness of the two components for RRAS. Without one of RDGANet and RGAT, we can not get better performance than RRAS. Consequently, the superiority of RRAS is due to both RDGANet for different relational representations and RGAT for efficient inter-agent interactions.



(a) Mean episode reward vs. number of updates in 8-agent coverage task.

(b) Mean episode reward vs. number of updates in 3 vs. 1 pursuit task

Fig. 10: Ablation studies regarding main components of RRAS.

Besides, to evaluate the performance of ablation methods with more agents, we further carry out an ablation study using curriculum learning in 20-agent coverage task. The evaluation results are shown in TABLE V. RRAS-w/o-RA is an ablation method that removes the relevant attention mechanism on the basis of RRAS. The results show that RRAS has the best performance compared with the other ablation methods. In particular, RRAS outperforms RRAS-w/o-RA, which shows the validity of removing invalid interactions among agents.

This ablation study further fully demonstrates the effectiveness of each component of the proposed RRAS.

TABLE V: Evaluation results in 20-agent coverage task.

Method	S%	MPR	MEL
RRAS-w/o-RDGANet	90.1 ± 1.54	-2.545 ± 0.0816	24.01 ± 0.129
RRAS-w/o-RGAT	94.3 ± 1.23	-2.226 ± 0.1039	21.20 ± 0.820
RRAS-w/o-RA	95.2 ± 0.23	-2.208 ± 0.0284	20.26 ± 0.894
RRAS (Ours)	**97.8** ± 0.12	**-1.894** ± 0.0839	**17.87** ± 0.260

V. Conclusion

In this paper, RRAS is proposed to learn the cooperative policies of distributed swarm systems for efficiently accomplishing specific tasks. In particular, RDGANet is designed to model different relational graphs among each agent and each type of entities for extracting different relational representations. RGAT is presented to cut off ineffective communication among irrelevant agents and model relevant communication topology for promoting efficient inter-agent interactions. Furthermore, the distributed actor-critic algorithm with full parameter sharing is implemented to learn cooperative swarm policies. Various simulation results validate the effectiveness and generalization of RRAS, and demonstrate that the proposed RRAS outperforms existing state-of-the-art methods on coverage and pursuit tasks.

References

[1] D. Yu, C. P. Chen, C.-E. Ren, and S. Sui, "Swarm control for self-organized system with fixed and switching topology," *IEEE transactions on cybernetics*, vol. 50, no. 10, pp. 4481–4494, 2019.

[2] Y. Liu, L. Wang, H. Huang, M. Liu, and C. Xu, "A novel swarm robot simulation platform for warehousing logistics," in *2017 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2017, pp. 2669–2674.

[3] R. D. Arnold, H. Yamaguchi, and T. Tanaka, "Search and rescue with autonomous flying robots through behavior-based cooperative intelligence," *Journal of International Humanitarian Action*, vol. 3, no. 1, 2018.

[4] S. Nag and L. Summerer, "Behaviour based, autonomous and distributed scatter manoeuvres for satellite swarms," *Acta Astronautica*, vol. 82, no. 1, pp. 95 – 109, 2013, 6th International Workshop on Satellite Constellation and Formation Flying.

[5] R. Lowe, Y. WU, A. Tamar, J. Harb, O. Pieter Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Advances in Neural Information Processing Systems 30*. Curran Associates, Inc., 2017, pp. 6379–6390.

[6] J. Jiang and Z. Lu, "Learning attentional communication for multi-agent cooperation," in *Advances in neural information processing systems*, 2018, pp. 7254–7264.

[7] T. Rashid, M. Samvelyan, C. S. De Witt, G. Farquhar, J. Foerster, and S. Whiteson, "Monotonic value function factorisation for deep multi-agent reinforcement learning," *The Journal of Machine Learning Research*, vol. 21, no. 1, pp. 7234–7284, 2020.

[8] M. Hüttenrauch, A. Šošić, and G. Neumann, "Guided deep reinforcement learning for swarm systems," *arXiv preprint arXiv:1709.06011*, 2017.

[9] M. Hüttenrauch, S. Adrian, G. Neumann *et al.*, "Deep reinforcement learning for swarm systems," *Journal of Machine Learning Research*, vol. 20, no. 54, pp. 1–31, 2019.

[10] C. Zhang, D. Song, C. Huang, A. Swami, and N. V. Chawla, "Heterogeneous graph neural network," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 793–803.

[11] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," *ArXiv*, vol. abs/1710.10903, 2017.

[12] A. Agarwal, S. Kumar, K. Sycara, and M. Lewis, "Learning transferable cooperative behavior in multi-agent teams," in *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, 2020, pp. 1741–1743.

[13] W. Wang, T. Yang, Y. Liu, J. Hao, X. Hao, Y. Hu, Y. Chen, C. Fan, and Y. Gao, "From few to more: Large-scale dynamic multiagent curriculum learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 05, 2020, pp. 7293–7300.

[14] Z. Pu, H. Wang, Z. Liu, J. Yi, and S. Wu, "Attention enhanced reinforcement learning for multi agent cooperation," *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[15] B. Zhu, L. Xie, D. Han, X. Meng, and R. Teo, "A survey on recent progress in control of swarm systems," *Science China Information Sciences*, vol. 60, no. 7, 2017.

[16] M. L. Littman, "Markov games as a framework for multi-agent reinforcement learning," in *Machine Learning Proceedings 1994*, W. W. Cohen and H. Hirsh, Eds. San Francisco (CA): Morgan Kaufmann, 1994, pp. 157 – 163.

[17] D. Busbridge, D. Sherburn, P. Cavallo, and N. Y. Hammerla, "Relational graph attention networks," *arXiv preprint arXiv:1904.05811*, 2019.

[18] H. Ryu, H. Shin, and J. Park, "Multi-agent actor-critic with hierarchical graph attention network." in *AAAI*, 2020, pp. 7236–7243.

[19] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Curran Associates, Inc., 2017, pp. 5998–6008.

[20] T. Zhou, F. Zhang, K. Shao, K. Li, W. Huang, J. Luo, W. Wang, Y. Yang, H. Mao, B. Wang *et al.*, "Cooperative multi-agent transfer learning with level-adaptive credit assignment," *arXiv preprint arXiv:2106.00517*, 2021.

[21] Y. Liu, W. Wang, Y. Hu, J. Hao, X. Chen, and Y. Gao, "Multi-agent game abstraction via graph attention neural network." in *AAAI*, 2020, pp. 7211–7218.

[22] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," *arXiv e-prints*, p. arXiv:1707.06347, Jul. 2017.

[23] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *Proceedings of the 26th Annual International Conference on Machine Learning*, ser. ICML '09. New York, NY, USA: Association for Computing Machinery, 2009, p. 41–48.

[24] T. Zhang, Z. Liu, S. Wu, Z. Pu, and J. Yi, "Multi-robot cooperative target encirclement through learning distributed transferable policy," in *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2020, pp. 1–8.